Tuomas Pihlajakoski

# WEB-SOVELLUKSEN TIETOTURVA

# WEB-SOVELLUKSEN TIETOTURVA

Tuomas Pihlajakoski
TIK8SNB
Opinnäytetyö
Syksy 2012
Tietojenkäsittelyn koulutusohjelma
Oulun seudun ammattikorkeakoulu

**TIIVISTELMÄ**

Oulun seudun ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma

---

Tekijä: Tuomas Pihlajakoski
Opinnäytetyön nimi: Web-sovelluksen tietoturva
Työn ohjaaja: Ani Ruusila
Työn valmistumislukukausi ja -vuosi: Syksy 2012 Sivumäärä: 43

---

Tässä opinnäytetyössä käsitellään web-sovelluksen tietoturvaa ensin teoriapohjalta ja sitten tietoa sovelletaan käytännössä ohjelmoitaessa osan Konehukka-internethuutokaupasta. Toimeksiantaja on O.P. Realisointipalvelu, joka on erikoistunut konkurssipesien ja rahoitusyhtiöiden omaisuuden noutoon, varastointiin ja realisointiin.

Työ on toteutettu PHP- ja MySQL-ohjelmoinnilla. Toteutuksen kannalta oleellista on asiakkaan palvelimelle asennetun PHP 4-version asettamat rajoitukset. Myös tietoturvan teoria on kirjoitettu siten, että sitä voidaan soveltaa PHP 4-versioon. Opinnätetyössä käydään läpi, kuinka ohjelmiston kehittäjä voi suojata ohjelman mm. XSS:ltä ja SQL-injektioilta. Työssä kerrotaan salasanojen tiivistämisen tärkeydestä ja sitten havainnollistetaan, kuinka tiivistäminen suoritetaan. Tietoturvaa on käsitelty myös tiedostojen siirtämisen ja eri käyttäjäryhmille näytettävien tietojen kannalta.

Aineisto koostuu suurimmaksi osaksi internet-lähteistä, koska internetistä löytyvät ajankohtaisimmat tiedot. Kirjallisia lähteitä on käytetty PHP:n ja MySQL:n perusteiden selittämiseen.

Työn lopputulos on osa internethuutokaupasta, joka on asennettuna toimeksiantajan palvelimelle. Huutokauppaa ei ole opinnäytetyön aikana tehty täysin valmiiksi, vaan sen kehittämistä on jatkettu vielä opinnäytetyön jälkeen.

---

Asiasanat: tietoturva, PHP, MySQL, HTML

**ABSTRACT**

Oulu University of Applied Sciences
Bachelor's degree, Business Information Systems

Author: Tuomas Pihlajakoski
Title of thesis: Web application security
Supervisor: Ani Ruusila
Term and year when the thesis was submitted: Autumn 2012 Number of pages: 43

The purpose of this thesis is to first discuss about the theory of web application security and then to implement it to the Konehukka internet auction site. The client of this thesis is O.P. Realisiointipalvelu which is specialized in the collection of property, realization and storage of bankruptcy estates and financing companies.

The application has been done using PHP and MySQL programming languages. There are some restrictions due to the fact that the client has PHP version 4 installed on their server. The theory has also been written to be compatible with PHP version 4. This thesis also contains information for programmers how to prevent XSS and SQL-injection attacks. It also covers the importance of password hashing and how to perform such operation. Safe file transfers and the outputting of data to certain user groups are also discussed.

In order to get the most recent and up to date data, the internet was used as the main resource. Some literary sources were used to explain the basics of PHP and MySQL.

The final product of this thesis was a part of the Konehukka internet auction site which was installed on the client's server. The product will be further developed and finished after completing this thesis.

Keywords: data security, PHP, MySQL, HTML

SISÄLTÖ

# 1 JOHDANTO

Tämän opinnäytetyön teoreettisessa osassa perehdytään web-sovelluksen tietoturvaan, jota hyödynnetään tehtäessä internethuutokauppaa O.P. Realisointipalvelulle. Tietoperustassa tutkitaan kirjallisten lähteiden avulla, mitä web-sovelluksen kehittäjän tulisi ottaa huomioon ja mitä ovat yleisimmät tietoturvan kannalta vaaralliset virheet tehtäessä uusia web-sovelluksia. Toiminnallisessa osiossa opittua tietoa hyödynnetään ohjelmoitaessa tarvittavat toiminnot internethuutokauppaan.

Opinnäytetyössä tehdään vain osa huutokaupasta ja sen kehittämistä jatketaan opinnäytetyön valmistumisen jälkeen. Tämän työn tuloksena huutokauppaan tehtiin valmiiksi tietokanta, käyttäjän rekisteröityminen, käyttäjän kirjautuminen ja tuotteen lisääminen.

Tietoturvaa käsitellään toimeksiantajan palvelimen ohjelmiston asettamien rajoitusten mukaan, joten tietoturvan teoriaosiossa esitetään menetelmiä, jotka ovat yhteensopivia PHP 4:n kanssa. Tämä sulkee pois joitakin uusimpia tietoturvamenetelmiä.

Työn toiminnallinen osio on tehty käyttäen PHP- ja MySQL -ohjelmointikieliä. PHP:n valintaan vaikutti erityisesti se, että se on suunniteltu web-sovelluksia varten. PHP:n etuna voi pitää myös sitä, että sitä voidaan käyttää useilla alustoilla ja eri käyttöjärjestelmillä.

Opinnäytetyön toimeksiantaja on O.P. Realisontipalvelu, joka on erikoistunut konkurssipesien ja rahoitusyhtiöiden omaisuuden noutoon, varastointiin ja realisointiin. Omaisuuden realisointi tapahtuu perinteisen huutokaupan avulla, suoralla myynnillä sekä internethuutokauppojen kautta. O.P. Realisointipalvelulla on jo olemassa omat internetsivut, joilla on ilmoitukset suoramyynnistä ja perinteisissä huutokaupoissa olevista tuotteista. Myöhemmin internethuutokauppaan voi tulla myös muita kohteita kuin pelkästään realisointipalvelun omia.

# 2 LÄHTÖKOHDAT

O.P. Realisointipalvelu tilasi internethuutokaupan opinnäytetyönä keväällä 2011, koska oma internethuutokauppa tulee huomattavasti edullisemmaksi kuin vuokrata se joltain kolmannen osapuolen paveluntarjoajalta. O.P. Realisiontipalvelu työllistää henkilöitä, jotka osaavat itse ohjelmoida, joten oma internethuutokauppa mahdollistaa myös haluttujen ominaisuuksien lisäämisen myöhemmin heidän niin halutessaan. Järjestelmän rakentamisessa ei käytetä mitään ulkopuolista sisällönhallintajärjestelmää, vaan se tehdään kokonaan itse, mikä tarkoittaa, että järjestelmä on alustariippumaton ja tietoturvaan on erityisesti kiinnitettävä huomiota.

## 2.1 Järjestelmän käsitteet

| KÄSITTEEN NIMI | KÄSITTEEN KUVAUS |
|---|---|
| Käyttäjä | Palveluun rekisteröitynyt henkilö tai yritys. |
| Ilmoitus | Tieto myytävästä tuotteesta |
| Kohdeyritys | O.P. Realisointipalvelu |
| Huuto | Käyttäjän tekemä tarjous ilmoitetusta kohteesta |
| Huutaja | Käyttäjä joka tekee tarjouksen ilmoitetusta kohteesta |
| Voittava huuto | Korkein kohteesta tehty tarjous ilmoitusajan umpeutuessa |
| Konkurssipesä | Konkurssiin ajautuneen yrityksen jäljelle jäänyt myytävä omaisuus. |
| Realisointi | Tapahtuu kun voittavan huudon omaava käyttäjä lunastaa huutamansa tuotteen. Tavaran muuttaminen rahaksi. |

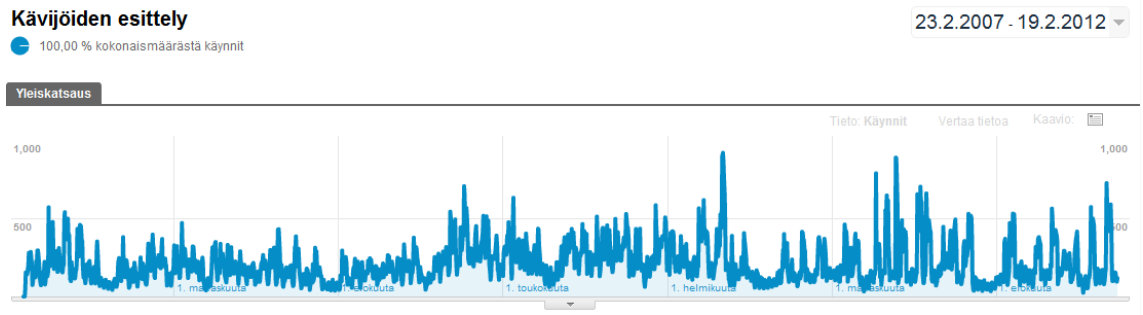
*Taulukko 1.Työn vaatimusmäärittelyssä nimetyt käsitteet.*

## 2.2 Järjestelmän kuvaus

Toteutettava kohdejärjestelmä on Internetissä toimiva huutokauppa-alusta. Alusta tulee toimimaan O.P. Realisointipalvelun myyntikanavana konkurssipesien ja rahoitusyhtiöiden koneille, laitteille sekä tavaroille. O.P. Realisointipalvelu lisää huutokaupattavat tuotteet alustalle ja palveluun rekisteröityneet käyttäjät tekevät niihin huutoja. Huutoajan umpeutuessa suurimman huudon tehnyt käyttäjä saa sähköpostilla tapahtumasta tiedon, joka sisältää maksuehdot ja -tavat. Muut huutokauppaan osallistuneet saavat sähköpostitse tiedon ilmoituksen sulkeutumisesta, sekä siitä, että he eivät olleet tehneet korkeinta tarjousta.

## 2.3 Statistiikka

Kaikkien kuvaajien sisältämä tieto on kerätty aikavälillä 23.2.2007 - 19.2.2012. Kuvaajat on saatu Google Analytics -palvelusta.




*Kuva 1. Kävijämäärän lisääntyminen.*

Kuvassa 1 on yleiskatsaus O.P. Realisointipalvelun sivuston kävijämäärään aikavälillä 23.2.2007 - 19.2.2012. Kuvaajasta käy ilmi, että kävijöiden määrä on ollut koko ajan kasvussa ja kävijöitä on ollut paljon heti sivuston aukaisuvaiheessa. Huomattava kasvu tapahtui kesäkuussa 2010, kun O.P. Realisointipalvelu otti käyttöön uuden sivupohjan. Sivustolla käy päivittäin satoja kävijöitä ja parhaimmillaan kävijämäärä kasvaa tuhansiin vuorokaudessa. O.P. Realisointipalvelu ei ole koskaan markkinoinut yritystään, lukuunottamatta joitain yrityslahjoja, joita ovat esimerkiksi kynät.

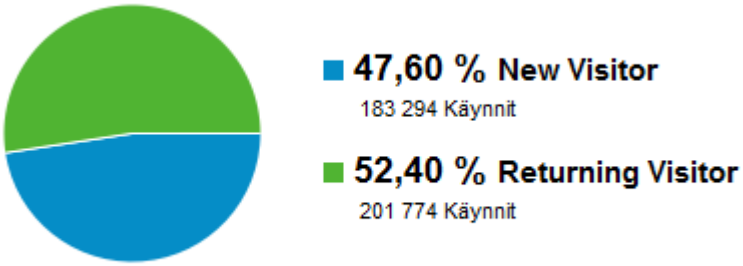
| | Kieli | Käynnit | % Käynnit |
|---|---|---|---|
| 1. | fi | 304 960 | 79,20 % |
| 2. | en-us | 44 269 | 11,50 % |
| 3. | fi-fi | 27 184 | 7,06 % |
| 4. | en | 2 494 | 0,65 % |
| 5. | sv | 2 120 | 0,55 % |
| 6. | en-gb | 1 561 | 0,41 % |
| 7. | sv-se | 1 164 | 0,30 % |
| 8. | ru | 468 | 0,12 % |
| 9. | et | 339 | 0,09 % |
| 10. | de | 81 | 0,02 % |

*Kuva 2. Selaimista saadut maatunnisteet.*

Kuvassa 2 näkyvät maatunnisteet, mistä maista on käyty O.P. Realisointipalvelun sivuilla. Poikkeuksellisen suuren "en-us" -määrän selittää osittain kävijän englanninkielinen internetselain, mutta siitä huolimatta O.P. Realisointipalvelu on harkinnut hankkia myös englanninkielisen version internethuutokaupastaan.


*Kuva 3. Kävijämäärät.*

Google analyticsin mukaan sivustolla on ollut noin 183 000 kävijää aikavälillä 23.2.2007 - 19.2.2012. Nämä kävijät ovat käyneet sivulla yhteensä noin 380 000 kertaa.



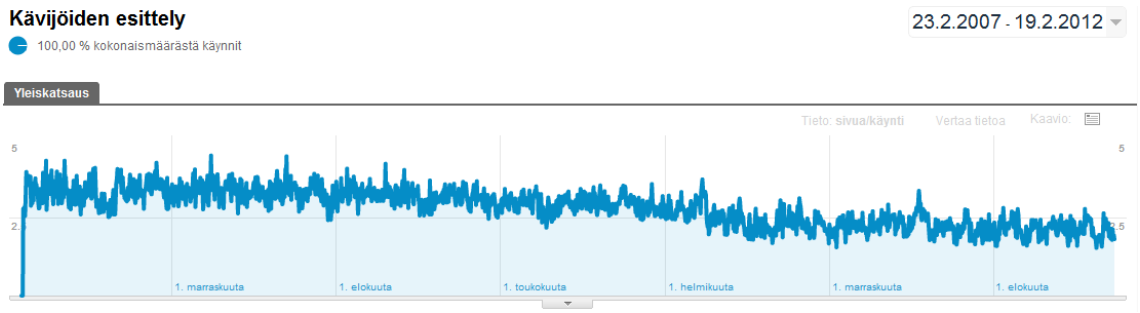

*Kuva 4. Uudet ja palaavat kävijät.*

Kuvan 4 kuvaajan mukaan 183 000 kävijästä 52,40 % on käynyt sivustolla uudestaan. Nämä 52,40 % ovat vastuussa runsaasta 380 000 käyntimäärästä.

| | Selain | Käynnit | % Käynnit |
|---|---|---|---|
| 1. | Internet Explorer | 226 656 | 58,86 % |
| 2. | Firefox | 128 194 | 33,29 % |
| 3. | Safari | 12 894 | 3,35 % |
| 4. | Chrome | 10 761 | 2,79 % |
| 5. | Opera | 5 470 | 1,42 % |
| 6. | Mozilla | 290 | 0,08 % |
| 7. | Mozilla Compatible Agent | 270 | 0,07 % |
| 8. | Android Browser | 161 | 0,04 % |
| 9. | Netscape | 133 | 0,03 % |
| 10. | Opera Mini | 51 | 0,01 % |

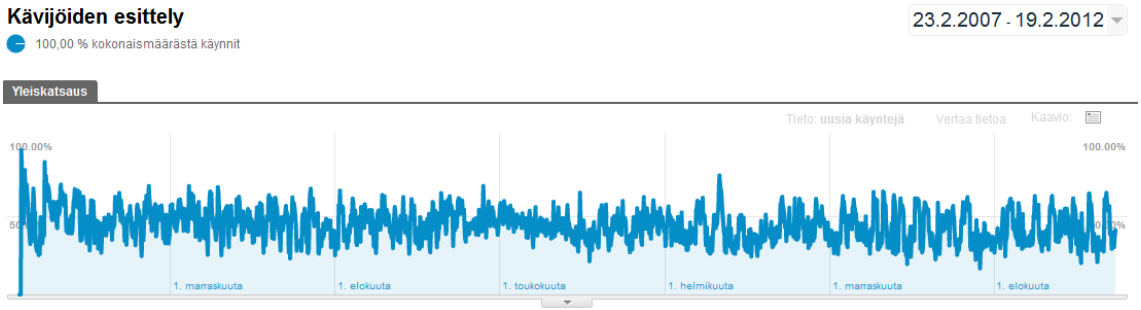
*Kuva 5. Selaimista saadut maatunnisteet.*

Kuvasta 5 käy ilmi, millä selaimilla sivustolla käydään. Internet Explorer ja Mozilla Firefox ovat olleet johtavina selaimina, joten huutokaupan pitää toimia niillä selaimilla hyvin.




*Kuva 6. Käynnit sivuilla / käynti.*

Kuvassa 6 näkyy kuinka monella sivulla yksittäinen kävijä on käynyt.




*Kuva 7. Kuvaajasta käy ilmi uusien kävijöiden suhde sivustolla aikaisemmin käyneisiin.*

Kuvan 7 kuvaajassa esitetään milloin sivustolle on tullut uusi kävijä suhteessa siihen, kuinka moni on sivustolla jo käynyt. Uusia kävijöitä on tullut tasaisesti ja noin puolet kävijöistä on jäänyt tai tulleet uudestaan sivustolle.

**Kustannukset**

O.P. Realisointipalvelun ilmoituskustannukset olivat vuonna 2010 yhteensä 48 678,41 €, joista lehti-ilmoitusten kustannukset olivat 7872,26 € ja netti-ilmoitusten kustannukset 40 806,15 €. Vuonna 2011 lehti-ilmoitusten kustannukset kasvoivat 10 362,72 €:oon, mutta netti-ilmoitusten kustannukset laskivat 40 291,58 €:oon. Kasvu lehti-ilmoitusten kustannuksista johtui osittain siitä, että niillä pyrittiin saamaan potentiaaliset asiakkaat vierailemaan realisointipalvelun kotisivuille.

# 3 YLEISET UHAT INTERNETISSÄ

PHP on laajasti käytetty yleisohjelmointikieli, joka on erityisen sopiva Web-sovellusten kehittämiseen ja jota voidaan upottaa HTML-kuvauskieleen (The PHP Group 2012b, hakupäivä 20.6.2012). PHP on pääosin palvelinohjelmointia, joten sillä voi tehdä kaikkea mitä millä tahansa muulla CGI-ohjelmalla voi tehdä, kuten noutaa tietoa lomakkeelta, luoda dynaamista sisältöä internet-sivuille tai lähettää ja vastaanottaa evästeitä (The PHP Group 2012b, hakupäivä 20.6.2012).

PHP on helposti opeteltava ohjelmointikieli ja monet ihmiset, joilla ei ole aikaisempaa kokemusta ohjelmoinnista, oppivat lisäämään sillä helposti interaktiivisuutta internetsivuilleen. Valitettavasti se usein tarkoittaa, että PHP-ohjelmoijat, etenkin aloittelevat, eivät tiedä mitä mahdollisia tietoturvariskejä heidän web-sovelluksissaan on (Child 2004, hakupäivä 13.6.2012).

Koskaan ei kannata luottaa siihen, että käyttäjät lähettävät sellaista tietoa, mitä sovelluksen kehittäjä olettaa heidän lähettävän. Tärkeää on muistaa myös, että kaikki tietoturva-aukkojen kautta vahinkoa tehneet käyttäjät eivät tee sitä tietoisesti tai tahallisesti. Web-sovellusta kehittettäessä on hyvä olettaa, että kaikessa käyttäjän lähettämässä datassa voi olla haitallista ohjelmakoodia, mukaan lukien jo asiakkaan käyttämässä sovelluksessa validoidussa koodissa. Tämä asia vaatii erityistä huomiota, jos web-sovelluksen tietoturvaa pitää tärkeänä (Child 2004, hakupäivä 13.6.2012).

Web-sovellukset koostuvat useista eri tekniikoista, tyypillisesti tietokantapalvelimesta, web-palvelimesta ja yhdestä tai useammasta ohjelmointikielestä, joista kaikkia edellämainittuja osia voidaan ajaa yhdellä tai useammalla käyttöjärjestelmällä. Siksi on tärkeää pysyä ajantasalla ja tehdä tarvittavat korjaustoimenpiteet ennen kuin joku käyttää haavoittuvuuksia hyväkseen (Apress Publishing 2010, hakupäivä 13.6.2012).

Data on yleensä elintärkeää yritykselle ja sen menettäminen voi olla tuhoisaa. Liian usein tietokantaa ja web-tilejä ei ole suojattu salasanoilla lainkaan tai salasanat ovat kyseenalaisia. Järjestelmänvalvojan sovelluksiin ei saa päästä käsiksi helposti tunnistettavan URL:n avulla. Tällaiset tietoturvavirheet eivät ole hyväksyttäviä, koska ne ovat helposti ratkaistavissa (Apress Publishing 2010, hakupäivä 13.6.2012).

## 3.1 SQL-injektio

SQL-injektio on perushyökkäys, jonka avulla joko pyritään saamaan aikaan luvaton pääsy tietokantaan, tai noutamaan tietoa suoraan tietokannasta. SQL-injektion periaatteet ovat yksinkertaiset ja tällaiset hyökkäykset ovat helppoja suorittaa (Kost 2004, hakupäivä 20.6.2012).

SQL-injektiot ovat luonteeltaan yksinkertaisia. Hyökkääjä syöttää merkkijonon sovellukseen tavoitteenaan manipuloida SQL-lausetta hyödykseen. Hyökkäyksen monimutkaisuuteen vaikuttaa se, että hyökkääjä ei välttämättä tiedä miten SQL-lause on muodostettu, eikä siten tiedä, miten sitä voi manipuloida (Kost 2004, hakupäivä 20.6.2012).

Yleisin SQL-injektiohyökkäys on SQL-manipulaatio. Hyökkääjä pyrkii muuttamaan olemassa olevaa SQL-lausetta lisäämällä ehtoja `WHERE` -ehtolauseeseen jatkamalla SQL-lausetta esimerkiksi `UNION`, `INTERSECT` tai `MINUS` -operaattoreilla. On olemassa muitakin mahdollisia variaatioita, mutta nämä ovat merkittävimmät esimerkit (Kost 2004, hakupäivä 3.9.2012). Seuraava lause ei ole täsmälleen sama kuin alkuperäisessä lähteessä, vaan sitä on muokattu selkeämmäksi.

```
SELECT tuote_nimi FROM kaikki_tuotteet
WHERE tuote_nimi like '%Tuolit%'
```

Lause palauttaa listauksen saatavilla olevista tuotteista. Hyökkääjä pyrkii muuttamaan lausetta siten, että se suoritetaan esimerkiksi seuraavalla tavalla:

```
SELECT tuote_nimi FROM kaikki_tuotteet
WHERE tuote_nimi like '%Tuolit'
UNION
SELECT tunnus FROM dba_tunnukset
WHERE tunnus like '%'
```

Lause ei ole täsmälleen sama kuin alkuperäisessä lähteessä, vaan sitä on muokattu selkeämmäksi. Listauksessa palautetaan kaikkien tuotteiden lisäksi myös tietokannan käyttäjätunnukset (Kost 2004, hakupäivä 4.9.2012).

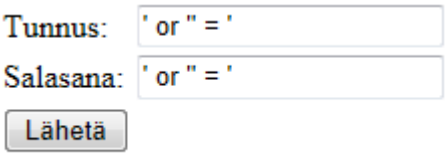
Yleensä SQL-manipulaatio suoritetaan kirjautumisen autentikoinnissa. Yksinkertainen web-sovellus saattaa todentaa käyttäjän oikeaksi suorittamalla SQL-lause ja tarkistamalla palauttaako se rivejä. Esimerkkilause ei ole täsmälleen sama kuin alkuperäisessä lähteessä, vaan sitä on muokattu selkeämmäksi.

```
SELECT * FROM users WHERE username = 'tunnus' and PASSWORD
= 'salasana'
```

Hyökkääjä pyrkii muokkaamaan lausetta siten, että se suoritetaan esimerkiksi seuraavalla tavalla:

```
SELECT * FROM users
WHERE username = 'tunnus' or '' = '' and PASSWORD =
'salasana' or '' = ''
```

Lause ei ole täsmälleen sama kuin alkuperäisessä lähteessä, vaan sitä on muokattu selkeämmäksi. Operaattorin arvojärjestyksen perusteella `WHERE`-ehtolause on aina totta ja hyökkääjä on saanut oikeiden sovellukseen (Kost 2004, hakupäivä 4.9.2012).

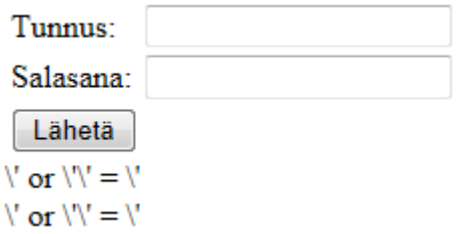



*Kuva 8. Injektio kirjoitettuna lomakkeeseen. Salasana näytetään normaalina tekstinä selkeyden vuoksi.*

Kuvassa esitetään, kuinka SQL-injektion voi suorittaa kirjautumislomakkeessa. Ensimmäinen heittomerkki sulkee alkuperäisen ehtolauseen, jonka jälkeen aloitetaan uusi ehto `OR`-operaattorilla. Heittomerkkien välissä ei ole mitään arvoa, joten se on tyhjä. Koska tyhjä arvo on aina yhtä suuri kuin tyhjä arvo, lause palauttaa yhden rivin. Yksi palautettu rivi voi tässä esimerkki tapauksessa kirjata käyttäjän sisään.

**Ehkäiseminen**

SQL-injektion voi ehkäistä käyttämällä `mysql_real_escape_string()` -funktiota. Käytettävä muuttuja laitetaan sulkujen sisään, esim. `mysql_real_escape_string($muuttuja);`. Funktio lisää kenoviivan heittomerkkien eteen, jolloin heittomerkit tulevat escapoiduiksi eikä syötettyä tekstiä siten enää lueta ohjelmakoodiksi.




*Kuva 9. Escapointi lisää kenoviivat heittomerkkien eteen tehden injektio-ohjelmakoodista tavallista tekstiä.*

## 3.2 Cross Site Scripting

Cross site scripting (XSS) -hyökkäykset ovat injektiohyökkäyksiä, joissa haitallista ohjelmakoodia syötetään muuten hyville ja luotettaville web-sivuille. Cross site scripting - hyökkäys tapahtuu, kun hyökkääjä käyttää web-sovellusta

lähettääkseen asiakasohjelmalla haitallista ohjelmakoodia eri loppukäyttäjille. Virheet, jotka sallivat näiden hyökkäysten onnistumisen, ovat aika yleisiä ja tapahtuvat siellä missä web-sovellukset sallivat käyttäjän syöttää dataa jota ei validoida ennen tulostamista (OWASP 2011, hakupäivä 23.7.2012).

Hyökkääjä voi lähettää Cross site script -hyökkäysellä haitallista ohjelmakoodia pahaa-aavistamattomalle käyttäjälle. Loppukäyttäjän selain ei voi tunnistaa, että ohjelmakoodi ei ole luotettavaa, joten se ajaa koodin normaalisti. Tämä johtuu siitä, että selain ei tunnista ohjelmakoodia haitalliseksi, vaan luulee sen tulevan luotettavasta lähteestä. Tällöin haitallinen koodi pääsee käsiksi evästeisiin, istuntoihin tai muihin arkaluonteisiin tietoihin, joita käyttäjän selain säilyttää ja joita käytetään kyseisellä sivustolla. Syötetyllä ohjelmakoodilla voidaan myös uudelleenkirjoittaa HTLM-sivuston ulkoasua (OWASP 2011, hakupäivä 23.7.2012).

**Tyypillinen hyökkäys**

Epäluotettava data tulee usein HTTP-pyynnöistä URL-parametrien, lomakekenttien, header-metodien tai evästeiden muodossa. Data, joka tulee tietokannoista, web-palveluista ja muista lähteistä, on usein epäluotettavaa tietoturvan näkökulmasta. Epäluotettavaa dataa pitäisi aina käsitellä olettaen sen sisältävän haitallista ohjelmakoodia. Dataa ei kannata lähettää mihinkään ennen kuin on tehty kaikki tarpeelliset toimenpiteet hyökkäysten estämiseksi ja neutraloimikseksi (OWASP 2012, hakupäivä 24.7.2012). Injektio on hyökkäystekniikka, jossa datan sekaan syötetään merkkejä, jotka aiheuttavat sen, että komentotulkki tulkitsee datan ohjelmakoodina (OWASP 2012, hakupäivä 24.7.2012).

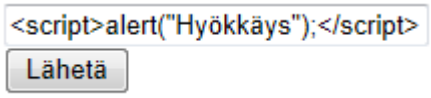
Esimerkki normaalista datasta:

```
<div>Data</div>
```

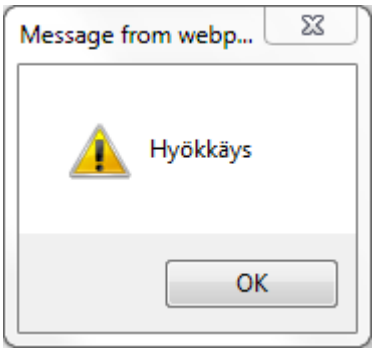
Esimerkki javascript-injektiosta datassa:

```
<div>Data<script>alert("Hyökkäys");</script></div>
```

Suojaamaton ohjelmakoodi syötetään lomakkeen tekstikenttään. Esimerkissä injektoidaan tavallisen datan sisään javascriptiä.




*Kuva 10. Javascript syötetään tekstikenttään kuten mikä tahansa normaali teksti.*




*Kuva 11. Huonosti toteutettu suojaus päästää javascriptin läpi sellaisenaan, jolloin ohjelmakoodi ajetaan ja esimerkin tapauksessa tuloksena on javascriptin `alert`-ilmoitus.*

**Ohjelmakoodin siistiminen (escaping)**

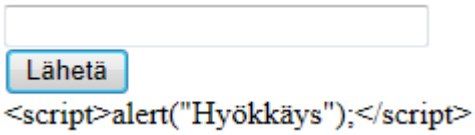
Siistimisen eli escaping-menetelmän tarkoitus on varmistaa, että merkkejä käsitellään datana, eikä merkkeinä, jotka ovat osa ohjelmointikieltä. Siistimistapoja on useita erilaisia. Escaping ei tarkoita, että dynaamista informaatiota käsiteltäisi informaation esittämisen yhteydessä. Jotkut näistä menetelmistä määrittelevät jonkin tietyn merkin suorittamaan siistimisen ja joissain muissa menetelmissä käytetään kehittyneempää syntaksia, johon liittyy useita eri merkkejä (OWASP 2012, hakupäivä 24.7.2012).

Escaping-menetelmä on ensisijainen tapa varmistaa, että epäluotettavaa dataa ei voi käyttää injektiohyökkäyksen välittämiseen. Datan perusteellisesta escapoimisesta ei ole mitään haittaa, vaan data esitetään selaimessa oikein. Komentotulkki tulkitsee escapoidun datan siten, että haitallista ohjelmakoodia ei suoriteta (OWASP 2012, hakupäivä 24.7.2012).

Yksi tapa ehkäistä XSS-hyökkäys, on laittaa lähdekoodissa suoritettavan datan eteen `htmlentities`-funktio (The PHP Group 2012d, hakupäivä 9.11.2012). Tällöin lähdekoodi voi näyttää esimerkiksi tältä:

```
$data = htmlentities($_POST['data']);.
```

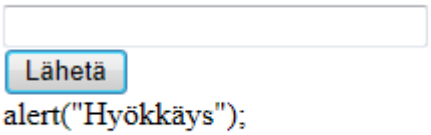
Käyttäjän syöttämä tieto tallennetaan `$data`-nimiseen muuttujaan, joka tulostetaan näkyviin sivulla. Sivuilla injektiokoodi tulostuu näin:




*Kuva 12. Htmlentities-funktio suodattaa injektio-ohjelmakoodin ja tulostaa sen normaalina tekstinä.*

Toinen yleinen tapa suodattaa data on käyttää `strip_tags`-funktiota. Se poistaa ylimääräiset HTML-merkit tulosteesta kokonaan (The PHP Group 2012e, hakupäivä 9.11.2012). Tätä tekniikkaa käyttäessä lähdekoodi voi saatta näyttää esimerkiksi tältä:

```
$data = strip_tags($_POST['data']);.
```




*Kuva 13. Strip_tags-funktio suodattaa injektio-ohjelmakoodin ja tulostaa sen kokonaan ilman HTML-tageja.*

## 3.3 Safe mode

Safe mode on poistettu kokonaan PHP 5.4:stä (The PHP Group 2012f, hakupäivä 9.11.2012). Safe mode varmistaa, että suoritettavan ohjelmakoodin

omistaja on sama kuin tiedoston, jota ohjelmakoodi yrittää avata (Developer shed 2010, hakupäivä 4.9.2012).

Safe moden tarkoitus oli toimia tietoturvamenetelmänä webhotelleissa, siinä missä virtuaalipalvelimet ja palvelimet, jotka ovat asiakkaan yksityisessä käytössä, eivät tarvinneet sitä. Safe moden pääasiallinen ongelma on se, että safe mode päällä jotkin perusfunktiot, joita jotkin web-skriptit käyttävät, eivät toimi (Server school 2010, hakupäivä 4.9.2012).

## 3.4 Salasanat

Salasanojen tiivistäminen on yksi yleisimpiä tietoturvan kannalta huomioon otettavia asioita, kun suunnitellaan sovellus, joka ottaa vastaan käyttäjien syöttämiä salasanoja. Ilman tiivistämistä mikä tahansa salasana, joka on tallennettu sovelluksen tietokantaan, voidaan varastaa, jos tietokantaan on murtauduttu. Salasanaa voidaan välittömästi käyttää sovellukseen tai muiden käyttäjien tilien ja muihin palveluihin murtautumiseen, jos salasanat eivät ole uniikkeja (The PHP Group 2012c, hakupäivä 25.9.2012).

Kun tiivistealgoritmia käytetään käyttäjän salasanaan ennen tietokantaan tallentamista, on epätodennäköisempää, että hyökkääjä voi päätellä salasanan, mutta samalla salasanatiivistettä voidaan verrata alkuperäiseen salasanaan myöhemmin. On kuitenkin tärkeää ottaa huomioon, että salasanojen tiivistäminen suojaa salasanoja vain tietokannassa, mutta ne voidaan silti siepata injektoimalla haitallista koodia sovellukseen (The PHP Group 2012c, hakupäivä 25.9.2012).

### 3.4.1 Sopimattomat tiivistealgoritmit

Tiivistealgoritmit kuten `md5`, `sha1` ja `sha256` on suunniteltu nopeiksi ja tehokkaiksi. Nykytekniikalla ja nykyaikaisella tietokonelaitteistolla on tullut tavalliseksi käyttää näiden algoritmien tulosteisiin ns. "brute-force"-hyökkäystä, jolla niiden alkuperäinen syöte saadaan selville. Koska nykyaikaiset tietokoneet

saavat helposti kumottua nämä tiivistealgoritmit, monet tietoturva-asiantuntijat eivät suosittele niiden käyttämistä salasanojen tiivistämiseen (The PHP Group 2012c, hakupäivä 25.9.2012).

### 3.4.2 Hyvät algoritmit

Kun salasanoja tiivistetään, on kaksi tärkeää asiaa, jotka pitää ottaa huomioon. Tietokoneen laskennalliset kustannukset ja ns. suola. Mitä enemmän tiivistealgoritmien laskennalliset kustannukset ovat, sitä kauemmin niiden murtaminen kestää. PHP:hen on paketoitu kaksi funktiota, `crypt()` ja `hash()`, joilla salasanan pystyy tiivistämään käyttäen määriteltyä algoritmia (The PHP Group 2012c, hakupäivä 25.9.2012).

#### Tietokoneen laskennalliset kustannukset

`Crypt`-funktio sisältää tuen useille tiivistealgoritmeille. Tätä funktiota käyttäessä käyttäjä voi olla varma, että valittu algoritmi on käytettävissä, koska PHP kykenee toteuttamaan jokaisen tuetun algoritmin vaikka käyttäjän oma järjestelmä ei pystyisikään (The PHP Group 2012c, hakupäivä 25.9.2012).

`Hash`-funktio tukee paljon enemmän muitakin algoritmeja ja variant-muuttujia kuin `crypt`-funktio, mutta se ei tue kaikkia algoritmejä joita `crypt`-funktio tukee (The PHP Group 2012c, hakupäivä 25.9.2012).

#### Suola

Salauksessa käytetään suolaa, joka on tiivistämisprosessissa käytettävää dataa, jonka tarkoitus on poistaa mahdollisuus katsoa tuloste esilasketuista tiivistepareista ja niiden syötteistä listassa, jota kutsutaan myös ns. rainbow-tauluksi (The PHP Group 2012c, hakupäivä 25.9.2012).

Suola on dataa, joka tekee tiivisteistä huomattavasti vaikeampia murtaa. Internetissä on useita eri palveluita, jotka tarjoavat paljon esilaskettuja tiivistelistoja, sekä alkuperäisen syötteen niihin tiivisteisiin. Suolan käyttäminen

tekee tiivisteen tuloksen löytämisen epätodennäköiseksi tai mahdottomaksi näistä listoista (The PHP Group 2012c, hakupäivä 25.9.2012).

Esimerkki suolan käyttämisestä `crypt`-funktion lisänä:

```
$salasana = 'salasana';
$crypt = crypt($salasana, 'suola');
```

Suolana on käytetty tavallista merkkijonoa, joka laitetaan `crypt`-funktiossa tiivistettävän salasanan jälkeen pilkulla erotettuna. Salasana tiivistetään ja voidaan tallentaa tietokantaan. Se saattaa näyttää tietokannassa esimerkiksi tältä:

```
su5WQa5Oprei6
```

## 3.5 Muuttujat

Php:ssä muuttujat esitetään siten, että ensimmäinen merkki on dollari ($), jota seuraa muuttujan nimi. Muuttujan nimi on merkkikokoriippuvainen (The PHP Group 2012a, hakupäivä 14.6.2012).

Muuttujan nimeäminen seuraa samoja sääntöjä kuin muutkin nimikkeet PHP:ssä. PHP:ssä kelpaava nimi alkaa kirjaimella tai alaviivalla, jota voi seurata muita kirjaimia, numeroita tai alaviivoja. Säännöllisenä lausekkeena se ilmoitettaisiin seuraavalla tavalla:

```
'[a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*'.
```

Poikkeuksena `$this` on erityinen muuttuja, jota ei voi itse määritellä (The PHP Group 2012a, hakupäivä 14.6.2012).

### 3.5.1 Globaalit muuttujat

Useissa ohjelmointikielissä täytyy luoda tietyt muuttujat, jotta niitä voidaan käyttää. PHP:ssä on mahdollista ottaa käyttöön `register_globals` -toiminto, joka mahdollistaa globaalien muuttujien käyttämisen ilman, että niitä tarvitsee itse luoda (Child 2004, hakupäivä 13.6.2012).

Seuraavaa esimerkkiä on muokattu selkeyden vuoksi:

```
if($salasana == "salasanani")
{
        $valtuutettu = 1;
}

if($valtuutettu == 1)
{
        echo "Tietoa vain valtuutetuille.";
}
```

Monille yllä oleva ohjelmakoodi voi näyttää hyvältä ja tätä samaa menetelmää käytetään kaikkialla internetissä. Kuitenkin, jos palvelimella on asetettu `register_globals` päälle, käyttäjän ei tarvitse kuin kirjoittaa osoiteriville `?valtuutettu=1`, jolloin hän pääsee käsiksi tietoon, johon hänellä ei normaalisti olisi valtuuksia. Tämä on yksi yleisimmistä PHP:n tietoturvaongelmista (Child 2004, hakupäivä 13.6.2012).

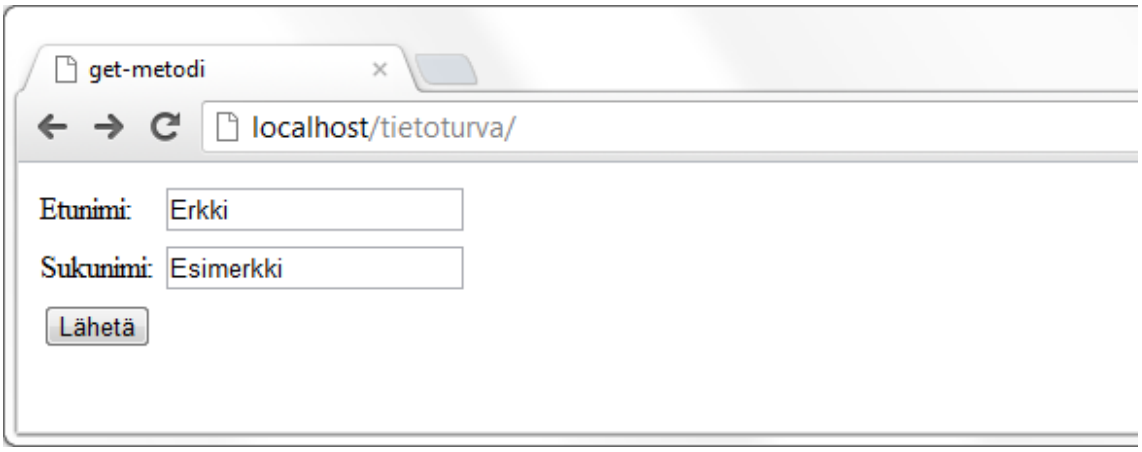
Ongelmaan on olemassa muutama ratkaisu, joista ensimmäinen ja ehkä paras on ottaa `register_globals` pois käytöstä. Toinen on varmistaa, että käytössä on vain ne muuttujat, jotka sovelluksen tekijä on määritellyt itse. Tämä tarkoittaa, että edellä olevan ohjelmakoodin alkuun lisättäisiin muuttujaksi: `$valtuutettu = 0;` (Child 2004, hakupäivä 13.6.2012).

### 3.5.2 GET-metodi

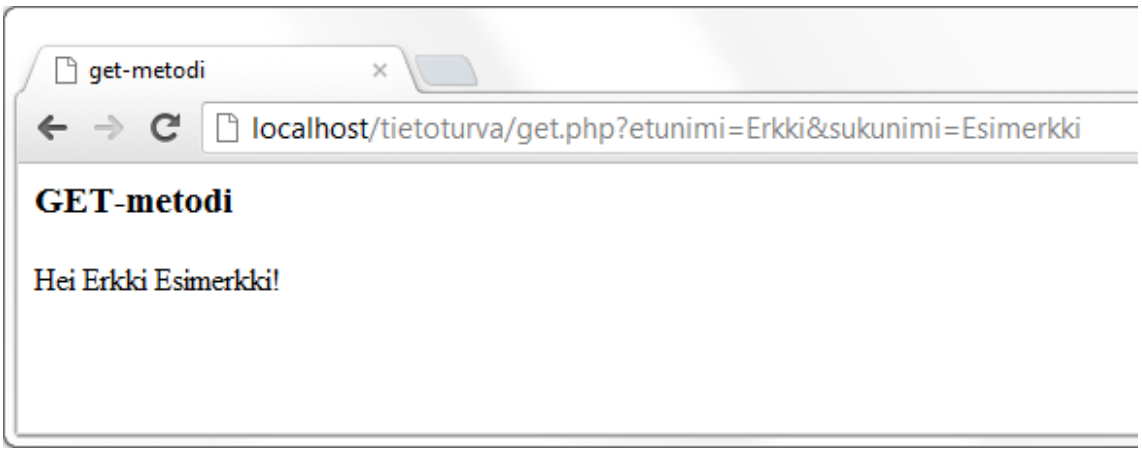
Ennaltamäärättyä `$_GET`-muuttujaa käytetään, kun lomakkeelta haetaan tietoa GET-metodilla. Tieto, joka on lähetetty GET-metodia käyttävällä lomakkeella, on näkyvissä kaikille selaimen osoitepalkissa ja sillä on rajoituksensa kuinka paljon tietoa sillä voidaan lähettää (w3schools 2012a, hakupäivä 10.9.2012).

GET-metodilla ei kannata koskaan lähettää arkaluonteista tietoa, kuten käyttäjätunnuksia ja salasanoja. Mutta, koska muuttujat näkyvät osoitepalkissa, voidaan sivusta tehdä kirjanmerkki. Tämä voi olla hyödyllistä joissain tilanteissa (w3schools 2012a, hakupäivä 10.9.2012).

Get-metodilla lähetettävä lomakkeen aloitustagi näyttää lähdekoodissa tältä: `<form action="get.php" method="get">`. Tagin sisällä oleva "action" lähettää lomakkeen tiedot lainausmerkkien sisällä olevaan osoitteeseen, jossa tiedot ajetaan ohjelmakoodin mukaisesti.




*Kuva 14. Lomake ennen tietojen lähettämistä.*

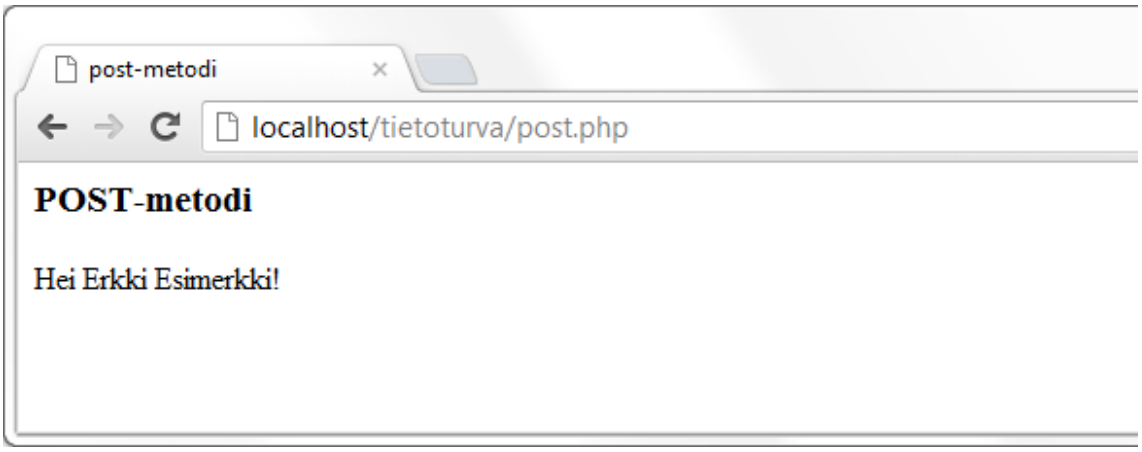



*Kuva 15. GET-metodilla lähetetyt tiedot.*

GET-metodilla palvelimelle lähetetyn lomakkeen tiedot näkyvät osoitepalkissa. Lomake lähettää tiedot uudelle sivulle, jonka nimi on "get.php".

### 3.5.3 POST-metodi

Ennaltamäärättyä `$_POST`-muuttujaa käytetään, kun lomakkeelta haetaan tietoa POST-metodilla. Tieto, joka on lähetetty POST-metodia käyttävällä lomakkeella, on näkymättömissä muilta eikä siinä ole rajoituksia datan määrän suhteen. Kirjanmerkkien lisääminen ei onnistu sivuilla, joiden data on lähetetty POST-metodilla (w3schools 2012b, hakupäivä 10.9.2012).




*Kuva 156. POST-metodilla lähetetyt tiedot. Osoitepalkissa ei näy muuttujia.*

POST-metodilla palvelimelle lähetetyn lomakkeen tiedot eivät näy osoitepalkissa, joten se soveltuu paremmin arkaluontoisen datan lähettämiseen, esimerkiksi käyttäjätunnus ja salasana ovat tällaista tietoa.

# 4 SOVELLUKSEN TOTEUTUS

Dynaamisten web-sivujen ohjelmointi ei onnistu samalla tavalla kuin staattisten HTML-sivujen tekeminen. Pelkän HTML-sivun testaaminen onnistuu helposti omalla tietokoneella avaamalla HTML-tiedosto selaimella. Suurinosa sovelluksen tiedostoista on kuitenki PHP-tiedostoja, jotka eivät aukea ilman pavelinta. Tiedostojen muokkaaminen ja toistuva tallentaminen pavelimelle on liian hidasta ja kömpelöä sekä palvelinta turhaan rasittavaa. Paras ratkaisu on asentaa kotitietokoneelle jokin web-sovelluskehitysympäristö, josta löytyy valmiina virtualisoituna web-palvelin, tietokantarajapinta ja PHP.

## 4.1 Työkalut

Työkalujen laatu on oleellinen osa projektin onnistumista. Ohjelmointityöhön löytyy runsaasti erilaisia ilmaisia työkaluja. Sovelluskehitysympäristötyökaluksi valinta kohdistui WampServer-ohjelmaan. Sen käyttöönotto osoittautui helpoimmaksi vaatien vähiten asetusten muokkaamista verrattuna esimerkiksi kilpailevaan XAMPP-ohjelmaan, jossa oli joitakin yhteensopivuusongelmia Windows 7:n kanssa projektin aloittamisen aikaan.

PHP:tä voi ohjelmoida millä tahansa tekstieditorilla, mutta ohjelmointia helpottaa kuitenkin käyttää jotain editoria, joka on samalla osa ohjelmointiympäristöä. Näistä NetBeans on ollut käytössä aikaisemmin ja sen toiminta ja suorituskyky on havaittu erinomaiseksi. NetBeans:n kaltaisen ohjelmointiympäristön käyttämisen etuja ovat mm. tiedostorakenteen näkyminen editorissa sekä useiden erilaisten liitännäisten runsaus. Liitännäinen voi esimerkiksi tarkistaa ohjelmakoodin syntaksin oikeellisuuden.

Web-sivuston suunnittelun on tehnyt kolmannen osapuolen graafikko. Vähäiseen kuvankäsittelyyn liittyviin tehtäviin sopii parhaiten ilmainen GIMP-sovellus. Siinä on useita samoja ominaisuuksia kuin sen kaupallisissa

vastineissa. Tärkeintä on kuitenkin, että se pystyy aukaisemaan Adobe Photoshop-tiedostoja.

## 4.2 Käyttäjätyypit

Huutokaupan toiminnallisuuden kannalta oleellista on kolme eri käyttäjätyyppiä. Ne ovat ylläpitäjä, asiakas ja myyjä, joista kullakin on omat yksilölliset käyttäjäoikeutensa.

**Ylläpitäjä**

Ylläpitäjällä on kolmesta käyttäjätyypistä eniten oikeuksia. Ylläpitäjän oikeuksia ovat uusien käyttäjien hyväksyminen ja heidän käyttöoikeuksien lisääminen ja poistaminen, sekä vanhentuneiden ilmoitusten poistaminen.

Toimeksiantaja vaati, että ylläpitäjälle tehdään oma tietokanta tietoturvasyistä, joten se käyttää kahta tietokantaa yhtäaikaisesti. Useamman tietokannan käyttäminen ehkäisee myös mahdollista käyttäjätilien sekoittumista. Tietoturvan lisäämiseksi ylläpitäjän kirjautumiseen ei tehdä sivustolle siihen ohjaavaa linkkiä, eikä ylläpitäjän kirjautumislomakkeella voi kirjautua tavalliselle tilille (SQLTeam, hakupäivä 9.11.2012).

**Asiakas**

Asiakkaaksi rekisteröidytään lomakkeella, jonka kaikki kentät pitää täyttää tai rekisteröityminen keskeytetään ja käyttäjälle ilmoitetaan hänen tekemät virheet. Oikein täytetyn lomakkeen tietojen lähettämisen jälkeen, asiakkaalle lähetetään automaattinen sähköpostiviesti, jonka esittämästä linkistä asiakas voi aktivoida käyttäjätilinsä.

**Myyjä**

Rekisteröityessään huutokaupan asiakkaaksi, on käyttäjällä mahdollisuus lähettää pyyntö, jolla hänet voidaan muuttaa myyjäksi. Myyjä voi tehdä uusia

ilmoituksia, sekä tehdä huutoja ilmoituksiin, jotka eivät ole hänen omiaan. Oman ilmoituksen huutaminen on estetty väärinkäytön vuoksi.

## 4.3 Rakenne

Palvelimella, jolle huutokauppa tehdään, on käytössä PHP 4, joka tukee olio-ohjelmointia, mutta päätin kuitenkin olla käyttämättä sitä, koska kyseessä on ensimmäinen iso PHP-projekti ja tavoitteena on ollut samalla oppia PHP:n ja MySQL:n perusteet. Ohjelmoinnissa on käytetty runsaasti valmiita ja itse tehtyjä funktioita, jolloin lähdekoodi pysyy siistinä ja ohjelmaa on helpompi muokata myöhemmin.

### Tietokanta

Tietokannan suunnittelu ja toteutus ovat ensimmäinen asia, jotka opinnäytetyöprojektin toiminnallisessa osiossa on tehty. Alun perin tietokanta on kirjoitettu kokonaisuutena sql-tiedostoksi ja ajettu phpMyAdmin MySQL-tietokannan hallintatyökalulla, mutta myöhemmin sovellusta ohjelmoitaessa tietokantaa on kuitenkin jouduttu muokkaamaan useasti phpMyAdminin avulla. Tietokannan ja siihen tallennetut tiedot saa ladattua helposti sql-tiedostoksi phpMyAdminilla, jos tietokannan haluaa siirtää esimerkiksi toiselle palvelimelle nopeasti.

### Tiedostorakenne

Tiedostot on jaoteltu loogisesti tiettyihin paikkoihin palvelimella. Tiedostot on jaettu omiin kategorioihinsa. Tiedostot, joilla näytetään asiakkaalle sovelluksen käyttöliittymä, ovat kaikki samassa paikassa juurihakemistossa.

Sivuston ulkoasu on tehty hyödyntäen CSS-tiedostoa ja se on omassa kansiossaan. Tyylit voi määritellä myös HTML-sivun alussa, mutta omassa tiedostossaan se on siistimpi ja selkeämpi tapa muokata ulkoasua, eikä tällöin tyylin muokkaamiseen käytetty koodi näy sivun lähdekoodissa (w3schools 2012c, hakupäivä 9.11.2012).

Sivuston toiminallisuuden kannalta erittäin tärkeitä ovat funktiot. Funktiot on tehty php-tiedostoihin, jotka on nimetty sisältämiensä funktioiden toimintojen mukaan. Ilmoituksien tekemiseen ja tulostamiseen liittyvät funktiot ovat PHP-tiedostossa, jonka nimi voi olla esimerkiksi "ilmoitus.funktiot.php". Ylläpitäjän, ilmoitusten, käyttäjien, kuvien sekä näytekuvien luontifunktioille on omat tiedostonsa.

Sivuston ulkoasussa käytettävät kuvat ovat omassa kansiossaan. Ne ovat erillään niistä kuvista, jotka ilmoituksen tekijä liittää ilmoitukseensa. Ilmoitusten mukana palvelimelle ladattaville kuville on oma kansio ja alikansio, johon tallennetaan pienennetyt näytekuvat.

Käyttöliittymän kannalta on olennaista, kuinka hyvin navigointi on toteutettu. Navigoinnille on tehty PHP-tiedosto omaan kansioonsa, josta se liitetään `include`-toiminnolla kohteeseensa.

**Sivun rakenne**

Sivun staattinen ulkoasu on jaettu kahteen php-tiedostoon, joita kutsutaan `include`-komennolla jokaisella sivulla, jotka näytetään käyttäjälle. Tällä menetelmällä säästetään aikaa ja vaivaa, kun tehdään jokin uusi sivu, eikä sivuille tarvitse erikseen kopioida HTML-koodia, jolloin vältytään mahdollisilta virheiltä. Tällä tavalla myös tiedostojen muokkaaminen on helpompaa (Tizag.com, hakupäivä 9.11.2012). Parhaiten tätä havainnollistetaan esimerkillä. Staattinen sivu on jaettu kahteen tiedostoon, joiden nimet ovat "header.php" ja "footer.php". Näistä ensimmäisen sisältämä ohjelmakoodi saattaa näyttää tältä:

```
html  head  meta
<!DOCTYPE HTML>
<html>
<head>
<title>Sivun yläosa</title>
<?php
header('Content-Type: text/html; charset=utf-8');
?>
<meta http-equiv="content-type" content="text/html; charset=UTF-8" />
<link href="css/tyylit.css" rel="stylesheet" type="text/css">
<link rel="shortcut icon" href="favicon.ico">
</head>
<body>
    <div id="navigointi">
        <?php
        include 'navigointi.php';
        ?>
    </div>
    <div id="sisalto">
```

*Kuva 167. Header.php-tiedoston sisältämä ohjelmakoodi.*

Huomioitavaa on, että `include`-toimintoa voidaan käyttää HTML-koodin sisällä laittamalla sen PHP-tagien sisään. Esimerkissä navigointi on tehty omaan tiedostoonsa, joka sitten liitetään div-elementin sisälle. Esimerkissä diviä, jonka id on "sisältö", ei suljeta ollenkaan tässä tiedostossa, vaan kyseisen divin sulkeva tagi on "footer.php"-tiedostossa. Toinen tiedosto, eli "footer.php" voi näyttää esimerkiksi tältä:

```
    </div><!-- Sisältö div -->
    <div id="alatunniste">
        <footer>© Tuomas Pihlajakoski 2012</footer>
    </div>
</body>
</html>
```

*Kuva 18. Footer.php-tiedoston sisältämä ohjelmakoodi.*

Ohjelmakoodissa päättyvä div, jonka id on "sisältö", suljetaan tässä tiedostossa. Näitä kahta PHP-tiedostoa voidaan käyttää esimerkiksi seraavalla tavalla jossakin muussa tiedostossa:

```php
<?php
include 'header.php';

// Jokin php:llä tehty ohjelma, joka tulee
// "sisältö" divin sisälle.

include 'footer.php';
?>
```

*Kuva 19. Kaksi tiedostoa on otettu käyttöön `include`-komennolla.*

Esimerkissä molemmat tiedostot on otettu käyttöön `include`-komennolla ja niiden väliin on kirjoitettu kommentti, jonka tilalla voisi olla esimerkiksi funktio tai jotain muuta PHP-ohjelmakoodia.

## 4.4 Suojautuminen käytännössä

### 4.4.1 Virheilmoitukset

Virheilmoitukset ovat hyödyllisiä sovelluksen kehittämisen aikana, koska ne kertovat ohjelmoijan mahdolliset virheet yksityiskohtaisesti. Virheilmoitukset kannattaa kuitenkin ottaa pois käytöstä, kun sovellus otetaan käyttöön palvelimella. Virheilmoitukset voivat paljastaa sovelluksen ohjelmakoodista tietoa henkilöille, joille kyseiset tiedot eivät kuulu. Virheilmoituksien asetuksia voi muuttaa palvelimen "php.ini"-tiedossa. Kehittämisen ajaksi tiedostoon on kirjoitettu rivi `error_reporting = E_ALL`, jolloin kaikki virheet ja varoitukset näytetään. Virheilmoitukset saa otettua pois käytöstä muokkaamalla "php.ini"-tiedostosta `display_errors = Off`.

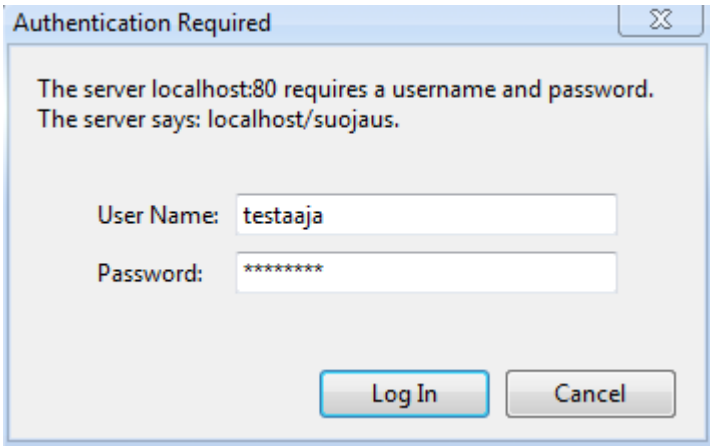
Sovelluksen ohjelmoija voi tehdä loppukäyttäjälle virheilmoituksia järjestelmän väärinkäytöstä. Ilmoitusten tarkoituksena on toimia ohjeena käyttäjälle, kuinka huutokaupan lomakkeita käytetään. Myös tietyt käsin kirjoitetut osoitteet, jotka esimerkiksi viittaavat käyttäjälle kiellettyihin toimintoihin, tulostavat sivulle virheilmoituksen.

### 4.4.2 .htaccess

Palvelimilla, joilla ajetaan Apache Web Server -ohjelmistoa, voidaan käyttää .htaccess-tiedostoa. Kun .htaccess-tiedosto on sijoitettu palvelimella johonkin kansioon, Apache Web Server tunnistaa ja suorittaa sen. Näillä .htaccess-tiedostoilla voidaan ottaa käyttöön tai kytkeä pois päältä toiminallisuuksia Apache Web Server -ohjelmiston asetuksia muokkaamalla. Näitä toiminnallisuuksia ovat mm. uudelleenohjaustoiminnot, sisällön salasanasuojaus tai kuvien suoralinkityksen estäminen.

**Salasanalla suojaaminen**

Tiedoston nimi on kokonaisuudessaan .htaccess, eikä sitä saa sekoittaa tiedostopäätteeksi. Tiedoston voi tehdä esimerkiksi Windows Notepadilla. Salasanalla suojaamista on käytetty sovelluksen kehittämisvaiheessa, kun on haluttu, että työn tuloksia on välillä siirretty myös palvelimelle asiakkaan nähtäväksi, eikä sinne ole haluttu vielä siinä vaiheessa päästää ulkopuolisia näkemään keskeneräistä huutokauppaa. Salasanasuojaus tulee esiin ponnahdusikkunana, jonka tekstikenttiin syötetään tunnus ja salasana. Ponnahdusikkuna saattaa näyttää esimerkiksi tältä:




*Kuva 170. Ponnahdusikkuna johon syötetään tunnus ja salasana.*

Palvelimelle luodaan kansio, jonka nimi on ".htpasswd" ja sen kansion sisään tiedosto, jonka nimi voi olla esimerkiksi "passwds". Kansiota ja sen sisältämää tiedostoa ei kannata luoda "public_html"-kansion sisään, koska tällöin se on sivuston hakemiston juuressa. Kansio kannattaa luoda tiedostohierarkiassa

"public_html"-kansion yläpuolelle, jolloin ulkopuoliset eivät pääse siihen käsiksi. Käyttäjätunnuksen ja salasanan sisältävä tiedosto voi olla nimeltään mikä tahansa, mutta kuvaava nimi on suositeltava.

Esimerkki .htaccess-tiedoston sisällöstä:

```
AuthType Basic
Authname "suojaus"
AuthUserFile "/home/tietoturva/.htpasswds/passwd"
require valid-user
```

*Kuva 181. .htaccess-tiedoston sisältö.*

Ensimmäisellä rivillä on autentikoinnin tyyppi, joka on esimerkissä `Basic`. Tämä tarkoittaa, että käytössä on HTTP-autentikointi. Toisella rivillä on nimetty kansio, jonka sisältö halutaan suojata. Kolmannella rivillä on polku tiedostoon, joka sisältää tunnuksen ja salasanan. Viimeisellä rivillä tarkennetaan, että vain voimassa oleva tunnus ja salasana ovat sallittuja.

Salasanan sisältävä tiedosto on sisällöltään yksinkertainen:

```
testaaja:$apr1$Hv9DBNkD$DXMmUXNSaSFwt58Br4zUw0
```

*Kuva 192. Passwd-tiedoston sisältö.*

Passwd-tiedoston sisälle kirjoitetaan tunnus ja salasana erotettuna toisistaan kaksoispisteellä. Salasanan voi kryptata käyttäen internetissä olevia htpasswd-salasanageneraattoreita.

### 4.4.3 Funktiot

PHP-kielinen ohjelma koostuu yhdestä tai useammasta ohjelmalohkosta, joita ovat pääohjelma ja funktiot (aliohjelmat). Funktio on pieni ohjelma, joka suorittaa tietyn rajatun tehtävän. Funktiot voidaan jakaa kirjastofunktioihin, jotka ovat kaikkien valmiina käytössä ja käyttäjän määrittelemiin funktioihin, joita jokainen ohjelmoija voi kirjoittaa omaan käyttöönsä. (Rantala 2005, 56.)

Funktion nimitys tulee siitä, että se palauttaa aina jonkin arvon. PHP:ssä arvo palautetaan `return`-lauseella, joka ei ole pakollinen. Jos return-lausetta ei käytetä, funktio palauttaa arvon `NULL`. Jos tätä arvoa ei käytetä, funktiota käytetään kuten aliohjelmaa (proseduuri). PHP sallii versiosta 4 alkaen funktioiden järjestyksen määrittelyn vapaasti: funktiot voivat olla esimerkiksi pääohjelmalohkon jälkeenkin. (Rantala 2005, 56.)

Sovelluksessa on käytetty runsaasti funktioita. Funktioiden sisälle on sovellettu teoriaosiossa läpi käytyjä tietoturvatekniikoita. Oleellisia tekniikoita ovat suodatukset, joita voi parhaiten demonstroida esimerkillä:

```
function rekisteröidy($nimi, $tunnus, $salasana)
{
    $nimi = mysql_real_escape_string(strip_tags($nimi));
    $tunnus = mysql_real_escape_string(strip_tags($tunnus));
    $suola = 'suola';
    $crypt = crypt($salasana, $suola);
    $sql_lause = //tiedot tallennetaan tietokantaan
}
```

*Kuva 203. Funktiossa käytetyt suodatukset.*

Kuvan funktiossa on käytetty `mysql_real_escape_string`- ja `strip_tags` -funktioita yhdistettynä suodattamaan sql-injektiot ja ylimääräiset HTML-tagit pois. Merkkijono "suola" tallenetaan muuttujaan, jonka nimi on `$suola`. Salasanan tiivistäminen tehdään `crypt`-funktiolla, jolle luodaan oma muuttuja, jonka nimi on `$crypt`. Funktion sisälle tulee tässä tapauksessa kaksi muuttujaa, joita ovat käyttäjän syöttämä `$salasana`, joka tiivistetään, sekä `$suola`, joka toimii tiivistämisen suolana. Lopulta suoritetaan SQL-lause, jolla tiedot tallennetaan tietokantaan.

## 4.5 Tiedostojen siirtäminen

Sovellus on tehty siten, että uutta ilmoitusta tehdessä käyttäjän on pakko lisätä siihen yksi kuva. Tämä tarkoittaa, että palvelimelle siirretään kuvatiedosto, johon viitataan tietokannassa. Lomakkeeseen täytyy lisätä `enctype`-attribuutti. Lomakkeen aloitustagi saattaa näyttää esimerkiksi tältä:

```
<form action="" method="post" enctype="multipart/form-data">
```

*Kuva 214. Enctype attribuutti lomakkeessa.*

Attribuutin arvoksi pitää kirjoittaa `multipart/form-data`, joka mahdollistaa tiedostojen siirtämisen.

Tiedoston lataaminen palvelimelle vaatii erityisten parametrien ja assosiatiivisen taulukon käyttämistä. Parametrejä ovat:

- `$_FILES['file']['name']` = ladatun tiedoston nimi
- `$_FILES['file']['type']` = ladatun tiedoston tyyppi
- `$_FILES["file"]["size"]` = tiedoston koko tavuina
- `$_FILES["file"]["tmp_name"]` = väliaikaisen palvelimelle tallennettavan kopion nimi
- `$_FILES["file"]["error"]` = lataamisesta johtuva virhekoodi

Kaikkia parametrejä ei ole kuitenkaan pakko käyttää.

Kyseessä on kuvan tallentaminen, joten sovelluksessa pitää olla ominaisuus, joka tunnistaa tallennettavan tiedoston tiedostopäätteen. Se voi esimerkiksi tulostaa virheilmoituksen ja estää ilmoituksen tallentamisen, jos tiedostopääte ei ole jpg tai png. Sallitut tietostopäätteet voi tallentaa taulukkoon esimerkiksi seuraavalla tavalla:

```
$sallitut_paatteet = array('jpg', 'png');.
```

Taulukko on tallennettu muuttujaan sen käyttämisen helpottamiseksi. Tiedostopäätteet voivat olla käyttäjällä isoilla kirjaimilla, joten sovelluksen on muutettava ne pieniksi. Kirjaimet voi muuttaa pieniksi `strtolower`-funktiolla, mutta tässä tapauksessa pitää kuitenkin käyttää useamman valmiin funktion yhdistelmää:

```
$tietostopaate = strtolower(end(explode('.', $kuvan_nimi)));
```

*Kuva 225. Tiedostopäätteeseen saa pienet kirjaimet usean funktion yhdistelmällä.*

Ensimmäiseksi tehdään `explode`-funktio, jonka sisään kirjoitetaan kaksi parametria. Näistä ensimmäinen on piste, joka toimii erottimena. Sen jälkeen tulee muuttujaksi tallennettu merkkijono, joka on tiedoston nimi kokonaisena. Tiedoston nimi voi olla esimerkiksi `testikuva.jpg`. Tiedostopääte on taulukon viimeinen osa, joten siihen päästään käsiksi `end`-funktiolla. Viimeiseksi tarvitaan vielä `strtolower`-funktiota, joka muuttaa muiden funktioiden ehdoilla haetun merkkijonon kirjaimet pieniksi. Tietokantaan tallennettaan kuvan tiedostopääte ja viittaus ilmoitukseen, johon kuva liitetään.

**Kansioiden tekeminen palvelimelle**

Palvelimelle pitää tehdä tarpeen mukaan yksi tai useampi kansio, kun sinne lisätään uusi kuva. Kansiot tehdään `mkdir`-funktiolla esimerkiksi:

```
mkdir('lataukset/'. $ilmoitus_id, 0744);
```

*Kuva 236. Kansiot tehdään `mkdir`-funktiolla.*

Funktion sisälle kirjoitetaan haluttu kansiopolku sekä kansiolle pavelimella annettavat käyttöoikeudet numeraalisena arvona. Lainausmerkkien sisällä on juurikansion nimi, jonka jälkeen on muuttujaksi tallennettu merkkijono, joka muuttuu tehtyjen ilmoitusten mukaan. Ensimmäiselle ilmoitukselle tehty kansiorakenne olisi "`lataukset/1/`". Numero kasvaa aina tehdessä uusi ilmoitus.

**Käyttöoikeudet**

Käyttöoikeudet määritellään neljän numeron sarjalla. Sarja aloitetaan sijoittamalla ensimmäiseksi numeroksi 0. Toinen numero määrittelee omistajan oikeudet. Kolmas numero määrittelee omistajan käyttäjäryhmän oikeudet. Neljäs numero määrittelee kaikkien muiden käyttäjien oikeudet.

- 0 = ei oikeuksia
- 1 = suoritus
- 2 = kirjoitus
- 3 = kirjoitus ja suoritus (1+2)
- 4 = luku

- 5 = luku ja suoritus (1+4)
- 6 = luku ja kirjoitus (2+4)
- 7 = luku, kirjoitus ja suoritus (1+2+4)

Esimerkin tapauksessa omistajalla on kaikki oikeudet (7). Omistajan käyttäjäryhmällä ja muilla käyttäjillä vain lukuoikeus (4).

**Kuvan siirtäminen palvelimelle**

Kuva siirretään palvelimelle `move_uploaded_file` -funktiolla sille tarkoitettuun yksilölliseen kansioon. Funktion sisälle tulee siirrettävän tiedoston nimi ja kohdesijainti. Funtio saattaa näyttää esimerkiksi tältä:

```
move_uploaded_file(
$image_temp, 'lataukset/'.$ilmoitus_id.'/'.$tiedosto_nimi.'.'.$tiedosto_paate
);
```

*Kuva 247. Kuvatiedoston siirtäminen palvelimelle.*

Esimerkissä siirrettävä tiedosto on väliaikainen palvelimelle tallennettu tiedosto, joka on saatu käyttämällä `$_FILES["file"]["tmp_name"]` -parametria. Pilkun jälkeen tulee kohdesijainti, joka koostuu merkkijoinoista ja muuttujista, joihin on tallennettu merkkijonoja.

## 4.6 Sisällön tulostaminen käyttäjille

Huutokaupan tietoturvan kannalta oleellista on, että ilmoitukset tulostetaan näkyvin oikein. Asiakas ja sivulla vierailevat käyttäjät saavat nähdä etusivulla olevan listauksen, jossa näytetään kaikki ilmoitukset niiden julkaisuajan mukaan. He saavat myös selata yksittäisen myyjän ilmoituksia. Myyjällä on samat oikeudet, mutta he saavat myös listata omat ilmoituksensa siten, että he näkevät myös suljetut ilmoituksensa. Ylläpitäjä saa käyttää omaa hallintasivustoa, johon ei ole kenelläkään muulla oikeutta päästä.

### 4.6.1 Julkinen sisältö

Etusivulle listataan kaikki ilmoitukset aikajärjestetyksessä, jolloin SQL-lause on erittäin yksinkertainen. SQL-lause on funktiossa, jota kutsutaan "index.php"-tiedostossa. Esimerkki, miten kaikki ilmoitukset listaavan SQL-lauseen voi muodostaa:

```
SELECT * FROM `konehukka`.`ilmoitus` WHERE
`ilmoitus_loppumiaika` > UNIX_TIMESTAMP() ORDER BY
`ilmoitus_id`;.
```

`SELECT *` -kyselylause hakee kaikki sarakkeet `konehukka`-tietokannasta `ilmoitus`-taulusta. `WHERE`-ehtolauseella haku rajataan niihin ilmoituksiin, joiden päättymisaika on isompi kuin tulostushetki, eli etusivulle tulostetaan näkyviin vain ne ilmoitukset, joissa on vielä jäljellä huutoaikaa. MySQL:ssä nykyisen kellonajan saa `UNIX_TIMESTAMP()`-funktiolla. Lopuksi listauksen järjestys määritellään ilmoituksen yksilöllisen id:n mukaan.

### 4.6.2 Henkilökohtainen sisältö

Huutokaupassa on paljon tilanteita, jolloin tarvitaan listausta, joka hakee tiedot tietyn yksilöllisen id:n mukaan. Tietoturvan kannalta on tärkeää suunnitella, mitkä toiminnot tätä menetelmää käyttävät. Esimerkiksi etusivun ilmoitusta painettaessa ilmoitus avautuu yksin omaan ikkunaansa, jolloin osoiteriville tulee näkyviin ilmoituksen id. Tätä id:tä voi käyttäjä muuttaa itse muokkaamalla sitä osoiterivillä. Ilmoitukset listaavalle sivulle pitää kirjoittaa PHP:llä lause, joka GET-metodin avulla hakee osoiterivilta ilmoituksen id:n. Sen voi tallentaa muuttujaksi esimerkiksi näin:

```
$ilmoitus_id = $_GET['ilmoitus_id'];.
```

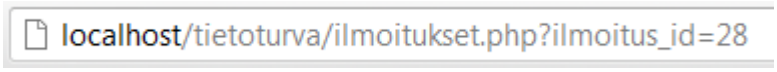
Linkki, jolla käyttäjä ohjataan ilmoitukseen, on muodossa:

```
<a href="ilmoitukset.php?ilmoitus_id=$ilmoitus_id">.
```

SQL-lause, jolla ilmoituksen id:n mukainen oikea ilmoitus saadaan näkyville, voi olla esimerkiksi tällainen:

```
SELECT * FROM `konehukka`.`ilmoitus` WHERE
`ilmoitus_loppumisaika` > UNIX_TIMESTAMP() AND
`ilmoitus_id`=".$_GET['ilmoitus_id'];.
```

`SELECT *` -kyselylause hakee kaikki sarakkeet `konehukka`-tietokannasta `ilmoitus`-taulusta. `WHERE`-ehtolauseella haku rajataan niihin ilmoituksiin joiden päättymisaika on isompi kuin tulostushetki. Tämä on käytössä tässäkin tilanteessa, koska se estää osoiterivillä haettavien vanhentuneiden ilmoitusten tulostamisen käyttäjälle. Ilmoituksen id saadaan GET-metodilla. Osoitepalkki saattaa näyttää esimerkiksi tältä:




*Kuva 28. Tarkasteltavan ilmoituksen id näkyy osoiterivillä, kun käytetään GET-metodia.*

# 5 POHDINTA

Työ oli erittäin mielenkiintoinen ja haastava. Suurin osa ajasta meni PHP- ja MySQL-ohjelmoinnin opettelemiseen ja projektin ohjelmointi alkoikin useasti kokonaan alusta ennen kuin olin tyytyväinen käytettyihin tekniikoihin. Sivuston staattista ulkoasua piti myös välillä muokata, johon meni jonkin verran ylimääräistä aikaa. Aihe on erittäin laaja ja siitä voisi kirjoittaa paljon enemmän kuin tässä opinnäytetyössä on käsitelty, joten raportin teoriaosan kirjoittaminen piti suunnitella huolellisesti, jotta liiallista haarautumista ei tulisi.

Sovellusten siirtyminen internettiin on lisääntynyt vuosien varrella huomattavasti. Web-sovellukset ovat syrjäyttäneet joitakin pikaviestimiä, esimerkiksi facebook on saanut useat siirtymään pois työpöytäohjelmien käytöstä. Lisääntyvät käyttäjämäärät vaativat ohjelmistojen kehittäjiltä aktiivista opiskelua uusien suojamenetelmien kehittämisestä (Andrew Valums 2010, hakupäivä 9.11.2012).

Aiheen valintaan vaikutti voimakkaasti halu oppia perusteet web-ohjelmoinnista. PHP:n ja MySQL:n valitsin niiden yleisyyden ja aloittelijaystävällisyyden vuoksi. Opinnäytetyö antoi hyvät valmiudet hakeutua alalle töihin ja jatkaa aiheen opiskelua töissä sekä vapaa-ajalla. PHP:n ja MySQL:n opettelu on helppoa runsaan kirjallisuuden ja internetlähteiden avulla. Erittäin hyväksi opiskelutavaksi havaitsin katsoa internetistä löytyviä selostettuja tutoriaaleja. Niissä yleensä tehtiin jokin pienimuotoinen sovellus alusta loppuun valmiiksi ja sitten sovelsin niistä opittuja asioita huutokauppaan.

Työn tietoturva tehtiin teoriassa esitettyjen menetelmien ohjeistamalla tavalla ja tietoturvaa on testattu yleisimmillä hyökkäysmenetelmillä. Testausprosessin aikana havaittiin myös, että käytetyt tietoturvamenetelmät eivät häiritse sovelluksen normaalia käyttöä. Jos käyttäjä syöttää hyväksyttyjä syötteitä lomakkeisiin, sovellus toimii yhtä sujuvasti kuin ilman minkäänlaisia tietoturvatoimintoja.

Sovelluksen kehittämistä jatketaan ja se saattaa laajentua kattamaan muitakin toimintoja huutokaupan lisäksi. Alustaa voi muokata haluamakseen, koska se on tehty alusta asti itse ja sillä saattaa olla muutakin kaupallista potentiaalia.

# 6 LÄHTEET


Andrew Valums. 2010. Hakupäivä 9.11.2012, http://valums.com/web-apps/

Apress publishing. 2010. Secure PHP Programming. Hakupäivä 13.6.2012, http://www.devshed.com/c/a/PHP/Secure-PHP-Programming.

Child, D. 2004. Writing Secure PHP. Hakupäivä 13.6.2012, http://www.addedbytes.com/writing-secure-php/writing-secure-php-1.

Developer shed. 2010 Hakupäivä 4.9.2012 http://www.devshed.com/c/a/PHP/Secure-PHP-Programming/1/

Hovi, A. 2000. SQL-ohjelmointi. Satku

Kost, S. 2004. An Introduction to SQL Injection Attacks for Oracle Developers. Hakupäivä 20.6.2012, http://www.net-security.org/dl/articles/IntegrigyIntrotoSQLInjectionAttacks.pdf.

OWASP. 2011. Hakupäivä 23.7.2012, https://www.owasp.org/index.php/XSS.

OWASP. 2012. Hakupäivä 24.7.2012, https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet.

Rantala, A. 2005. Web-ohjelmointi. Docendo

Server school. 2010. Hakupäivä 4.9.2012, http://www.serverschool.com/dedicated-servers/what-is-php-safe-mode/

SQLTeam. 2000. Hakupäivä 9.11.2012, http://www.sqlteam.com/article/single-database-or-multiple-databases

The PHP Group. 2012a. Hakupäivä 14.6.2012, http://www.php.net.

The PHP Group. 2012b. Hakupäivä 20.6.2012,
http://www.php.net/manual/en/intro-whatcando.php.

The PHP Group. 2012c. Hakupäivä 25.9.2012,
http://php.net/manual/en/faq.passwords.php.

The PHP Group. 2012d. Hakupäivä 9.11.2012,
http://php.net/manual/en/function.htmlentities.php

The PHP Group. 2012e. Hakupäivä 9.11.2012,
http://php.net/manual/en/function.strip-tags.php

The PHP Group. 2012f. Hakupäivä 9.11.2012,
http://php.net/manual/en/features.safe-mode.php

Tizag.com. 2008. Hakupäivä 9.11.2012, http://www.tizag.com/phpT/include.php

w3schools. 2012a. Hakupäivä 10.9.2012,
http://www.w3schools.com/php/php_get.asp

w3schools. 2012b. Hakupäivä 10.9.2012,
http://www.w3schools.com/php/php_post.asp

w3schools. 2012c. Hakupäivä 9.11.2012,
http://www.w3schools.com/css/css_intro.asp